ภาษาไทยในกราฟิกโหมด

เอาต์พุต

บุญเลิศ เอี่ยมทัศนา

การแสดงผลในแบบกราฟิกโหมด โดยเฉพาะบนเครื่องไมโครคอมพิวเตอร์นับวันจะมีความสำคัญและความจำเป็นมากขึ้น จนอาจกล่าวได้ว่าในอนาคตอันใกล้นี้การแสดงผลในแบบเท็กซ์โหมดคงหมดไป

ผมจึงขอนำเรื่องราวของการแสดงผลภาษาไทยในแบบกราฟิกโหมดบนเครื่องสิบหกบิต ตั้งแต่การสร้างตัวอักษร การอ่านคีย์ ไปจนถึงการแสดงผลบนจอภาพมากล่าวถึง ซึ่งก็ต้องขอเรียนให้ทราบว่าไม่ยากครับ

การแสดงภาพอักขระในกราฟิกโหมด

การแสดงภาพอักขระในกราฟิกโหมดเป็นการดำเนินการทางด้านซอฟต์แวร์ล้วน ๆ แตกต่างกับในเท็กซ์โหมดที่ต้องอาศัยฮาร์ดแวร์ (เช่น คาแรกเตอร์เยนฯ) เข้าช่วย จึงมีความอ่อนตัวกว่ากันมาก เช่น ใน 1 จอภาพจะให้มีกี่บรรทัดก็ได้ และไม่ขึ้นกับการ์ด หากการ์ดนั้นสามารถแสดงผลในแบบกราฟิกได้เป็นใช้ได้

แต่ในการเขียนโปรแกรมจำเป็นต้องขึ้นกับการ์ดอยู่บ้าง ทั้งนี้เพราะการ์ดแต่ละแบบมีจำนวนจุดภาพต่อ 1 เส้นสแกนไลน์ไม่เท่ากัน และจำนวนเส้นสแกนไลน์ต่อ 1 จอภาพไม่เท่ากัน ในบทความนี้จะถือการ์ดเฮอร์คิวลิสเป็นหลักเพราะมีผู้ใช้มาก แต่ถึงอย่างไรก็ตาม สำหรับหลักการแล้วสามารถใช้กับการ์ดได้ทุกแบบ

วิธีแสดงภาพอักขระสามารถดำเนินการได้ 2 แบบคือ ในแบบเวคเตอร์ดังที่กล่าวในบทความของผมในไมโครฯ ฉบับที่ 50 กับแบบบิตแมพซึ่งจะกล่าวและใช้ในบทความชุดนี้ ทั้งนี้เพราะการแสดงภาพอักขระในแบบเวคเตอร์ไม่เหมาะสำหรับงานที่ต้องการแสดงข้อความหลาย ๆ บรรทัด เพราะแสดงได้ช้า เหมาะกับงานทางด้านวาดเขียนมากกว่า

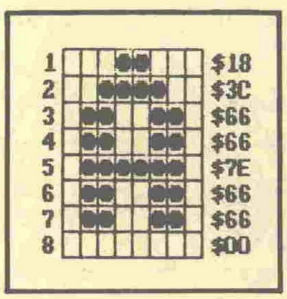
แพตเทิร์น บิตแมพ และคาแรกเตอร์เยนฯ

แต่ก่อนที่จะเดินเรื่องต่อไปผมขอทำความเข้าใจในถ้อยคำสามคำนี้คือ แพตเทิร์น (pattern) บิตแมพ (bitmap) และคาแรกเตอร์เยนฯ (charater-generator) ว่าผมนำมาใช้ในความหมายใดก่อนครับ กล่าวคือ

ในรูปที่ 1 ที่มองเห็นเป็นภาพอักขระ A คือแพตเทิร์นของอักษร A ขนาด 8×8 ซึ่งถ้าเราให้ค่า 1b (b หมายถึงเลขฐานสอง) แทนจุดดำในภาพ (ซึ่งจะเป็นจุดสว่างบนจอภาพ) และ 0b แทนจุดขาวในภาพ (ซึ่งจะเป็นจุดมืดบนจอภาพ) และถ้าเราใช้หน่วยความจำ 1 ไบต์ต่อ 1 แถว จะได้ค่าในหน่วยความจำแต่ละไบต์ดังแสดงเป็นเลขฐานสิบหก ($ หมายถึงเลขฐานสิบหก) ไว้ทางขวาของแต่ละแถว

เช่นในแถวที่ 1 (ตัวเลขทางซ้ายหมายถึง แถวที่และไบต์ที่ด้วย) เมื่อนำเลขฐานสองนี้ทางขวามาเขียนกลับเป็นเลขฐานสองจะได้เป็น

00011000



รูปที่ 1 แพตเทิร์นของอักษร A ในแบบ 8×8

ผมจึงขอเรียกหน่วยความจำทั้ง 8 ไบต์นี้ว่า **แพตเทิร์นของอักษร A** ด้วยเพราะเป็นแพตเทิร์นเช่นกัน แต่เป็นแพตเทิร์นในรูปของเลขฐานสอง (ความจริงไม่ใช่คำใหม่ ผมนำมากล่าวเพื่อให้ชัดขึ้นเท่านั้น) และขอรวมเรียกแพตเทิร์นของอักขระทั้งหลาย เช่น ในภาษาอังกฤษคือตั้งแต่รหัส ASCII 0 ถึง 127 และอาจต่อไปถึง 255 ว่าแพตเทิร์นของอักขระ หรือแพตเทิร์น

แพตเทิร์นนี้เมื่อสร้างแล้ว ในทางเท็กซ์โหมดจะนำไปบรรจุในหน่วยความจำ ROM แล้วเรียกไอซีตัวนี้ว่า **คาแรกเตอร์เยนฯ** คือตัวผลิตภาพอักขระ เพราะเมื่อประกอบกับวงจรอื่น ๆ อีกมันจะเป็นตัวทำให้เกิดภาพอักขระบนจอภาพ

ส่วนทางด้านกราฟิกโหมด เราจะเก็บแพตเทิร์นนี้ไว้ในหน่วยความจำ RAM หรือ ROM ก็ได้ดังเช่น ในเครื่อง IBM-PC เก็บไว้ในหน่วยความจำที่เรียกว่า ROM-BIOS แต่เทอร์โบปาสคาลใช้แพตเทิร์นของตนเอง ซึ่งจะโหลดลงหน่วยความจำ RAM เมื่อใช้คำสั่ง InitGraph

สำหรับแพตเทิร์นใน ROM-BIOS หนังสือภาษาอังกฤษบางเล่มเรียกว่าคาแรกเตอร์เยนฯ ซึ่งเป็นการเรียกโดยอนุโลม ในบทความของผมนี้จึงขอเรียกเป็น แพตเทิร์นในคาแรกเตอร์เยนฯ กับแพตเทิร์นใน BIOS เพื่อให้แตกต่างกัน และถือว่าในกราฟิกโหมดไม่มีคาแรกเตอร์เยนฯ เพื่อไม่ให้สับสน

ดังกล่าวแล้วว่าการแสดงภาพอักขระในกราฟิกโหมดจะต้องดำเนินการด้วยซอฟต์แวร์ ได้เขียนโปรแกรมเพื่อดำเนินการดังในรูปที่ 2 ซึ่งจะนำภาพอักขระใดก็ได้ที่ผ่านค่าแพตเทิร์นของมันมาให้แก่ตัวแปร apat ออกแสดงบนจอภาพ ณ ตำแหน่งที่อ่านค่าให้แก่ xx และ yy และเรียกการแสดงภาพในแบบนี้ว่า **บิตแมพ** และก็เรียกแพตเทิร์นว่าบิตแมพด้วย ซึ่งผมเองก็ไม่แน่ใจว่าเรียกอะไร ตามอะไร ในบทความนี้จึงขอแยกออกจากกัน

อีกเรื่องหนึ่ง คือ แพตเทิร์นของอักขระนี้ เช่น แพตเทิร์นของอักษรในภาษาอังกฤษ อาจทำไว้หลาย ๆ แบบ เช่น เป็นแบบ Triplex และ Gothic จะเรียกว่าเป็นฟอนต์ (font) คือเป็นฟอนต์แบบต่าง ๆ

กรอบอักขระ และจุด PP

การแสดงภาพในแบบเท็กซ์โหมดนั้นหน่วยภาพที่เล็กที่สุดจะแสดงได้คือ 1 อักขระ และเราไม่สามารถจะให้แสดงภาพอักขระเหลื่อมทับกันได้ จึงเปรียบเสมือนว่า ได้แบ่งจอภาพเป็นช่องอักขระเล็ก ๆ จำนวนมาก เช่น เป็น 80 ช่องต่อบรรทัด 25 บรรทัดต่อจอภาพ คือ 2000 ช่องอักขระต่อจอภาพ

สำหรับทางด้านกราฟิก หน่วยภาพที่เล็กที่สุดคือ 1 จุดภาพเช่นกัน เราไม่สามารถแสดงภาพสองจุดภาพให้ซ้อนทับกันในแบบเหลื่อมได้ แต่สามารถแสดงภาพอักขระให้เหลื่อมทับกันได้ เพราะอักขระหนึ่ง ๆ ประกอบด้วยหลายจุดภาพ กล่าวคือ ในกราฟิกโหมดถึงแม้จะไม่แบ่งจอภาพเป็นช่องอักขระ แต่การแสดงภาพอักขระหนึ่ง ๆ จะต้องอยู่ในกรอบอันหนึ่งเช่นกัน ดังรูปที่ 1 ประกอบกับโปรแกรมในรูปที่ 2 จะเป็นการแสดงภาพอักขระในกรอบขนาด 8×8 จุดภาพ โดยมีจุดอ้างอิงอยู่ตรงมุมบนซ้ายของกรอบอักขระ นั่นคือ เราสามารถกำหนดให้กรอบอักขระเหลื่อมทับกันได้ ปรากฏเป็นตัวอักษรซ้อนทับหรือเหลื่อมกันบนจอภาพ แต่ไม่ใช่เป็นวิธีแสดงผลโดยปกติ ที่นำมากล่าวเพื่อแสดงให้เห็นว่าทำได้ และจะใช้คำว่า **กรอบอักขระ** ต่อไป

การแสดงภาพอักขระในเท็กซ์โหมดใช้ตำแหน่งของเคอร์เซอร์เพื่ออ้างอิงว่าจะแสดงภาพอักขระในช่องอักขระใด ทางด้านกราฟิกก็เช่นกันมีจุด CP (Current Position) ซึ่งนอกจากเราจะสามารถเคลื่อนย้ายจุด CP ไปได้ทั่วทั้งจอภาพแล้ว (สำหรับการ์ดเฮอร์คิวลิส 720 จุดในแกน x และ 348 จุดในแกน y) เราสามารถจะให้จุด CP อยู่ตรงส่วนใดของกรอบอักขระก็ได้ เช่น มุมบนซ้าย มุมล้างซ้ายหรือขวา ซึ่งเรื่องหลังนี้ขึ้นอยู่กับโปรแกรมของเรา

แต่ในการเขียนโปรแกรมของเรื่องนี้ต้องการเขียนโปรแกรมตัวอย่างด้วยภาษาปาสคาล โดยใช้เทอร์โบปาสคาล หมายถึงว่าจะต้องใช้ระบบกราฟิกของบอร์แลนด์ คือ BGI (Borland Graphics Interface) ซึ่งใน BGI อ้างอิงถึงจุด CP อยู่แล้ว แต่ผมจะไม่ใช้จุดนี้และจะใช้คำในเทอร์โบปาสคาลเฉพาะเท่าที่สามารถแปลงเป็นภาษาแอสเซมบลีได้โดยตรง (ไม่ใช่ย้อนกลับไปกับมาดังเช่นจะเป็นเมื่ออ้างถึงจุด CP)

ทั้งนี้ เพราะผมเห็นว่าโปรแกรมนี้ควรอยู่ในระดับระบบจัดการเมื่อนำไปใช้จริงจึงขอเปลี่ยนเรียกชื่อจุดนี้เป็น PP (Present Position) แทน และต่อไปจะให้ตัวแปร prex กับ prey เก็บตำบลที่ของจุดนี้

ปัญหาในภาษาอังกฤษ

ผมว่าในโลกนี้เต็มไปด้วยปัญหาทั้งนั้นครับ ในภาษาอังกฤษที่เราเห็นว่าเขาไม่มีปัญหา ความจริงแล้วก็มีปัญหาเช่นกัน เช่นในยุคที่จอภาพมี resolution ต่ำ ช่องอักขระหนึ่ง ๆ จะมีเพียง 8×8 จุดภาพเท่านั้น และก็แสดงภาพอักขระไม่ได้เต็มช่องอักขระด้วย

เพราะต้องเว้นที่สำหรับอักษรที่ติดกันทั้งทางข้างและคนละบรรทัด ส่วนมากจึงให้อักขระอยู่ในช่อง 7×7 จุดภาพ ก็มีปัญหาเรื่องหางกับหัว เช่น หางตัว g กับ h ว่าจะให้ส่วนใดยาวมากน้อยเท่าไร

ต่อมา ในยุคของ IBM-PC จอโมโนโครมมี resolution สูงขึ้น ทำให้สามารถขยายช่วงอักขระได้เป็น 9×14 จุดภาพ โดยที่แพตเทิร์นของอักขระหนึ่ง ๆ จะเป็น 8×14 บิต ทำให้ได้ภาพที่สวยงามขึ้นในเท็กซ์โหมด

ทางด้านกราฟิกโดยที่ไอบีเอ็มใน

```
TYPE   PatternT = ARRAY[0..7] OF Byte;

PROCEDURE  PutBitMap(apat : PatternT);
TYPE    BitMapT = ARRAY[0..7] OF Byte;
CONST   bits : BitMapT = ($80,$40,$20,$10,
                          $08,$04,$02,$01);
VAR     xx,yy,x,y,cnt : Word;
BEGIN
  xx := GetX;  yy := GetY;
  for  y := 0 to 7  do
    for  x := 0 to 7  do
      if  apat[y] and bits[x] <> 0  then
        PutPixel(xx+x,yy+y,GetMaxColor);
END;
```

รูปที่ 2 โพรซีเดอร์ที่จะนำแพตเทิร์นในแบบ 8×8 ไปออกจอภาพ

ตอนแรกเพ่งเล็งเฉพาะกราฟิกบนการ์ด CGA ซึ่ง resolution ต่ำ แพตเทิร์นใน BIOS จึงเป็นแพตเทิร์นแบบเก่าสำหรับกรอบอักขระขนาด 8×8 จุดภาพ

โปรแกรมกึ่งสำเร็จรูปที่สร้างสำหรับจอภาพที่มี resolution สูงไม่ว่าจะเป็นโมโนโครม หรือสี จะสร้างแพตเทิร์นของตนเอง ซึ่งถึงแม้จะเป็นขนาด 8×8 บิตเท่ากัน (เพราะไม่มีความจำเป็นใดที่จะต้องใช้หน่วยความจำมากกว่านี้) ดังแสดงเปรียบเทียบให้ดูในรูปที่ 3 แต่ใช้วิธีย้ายจุด CP ทำให้หางตัว Q ห้อยลงมาข้างล่าง ได้ภาพอักขระที่สวยงามกว่า

ปัญหาภาษาไทย

มีเรื่องที่ผมไม่ได้เรียนให้ทราบแต่แรกว่า การทำภาษาไทยทำได้หลายแบบ ที่ผมนำมาแสดงเป็นเพียงแบบหนึ่งเท่านั้น และจะทำไปถึงจุดหนึ่งเทียบได้กับคำในภาษาปาสคาลว่า Read กับ Write โดยใช้ชื่อว่า ReadTStr กับ WriteTStr เพราะจะดำเนินการกับสตริงเท่านั้น

ปัญหาของภาษาไทยมีมากกว่าภาษาอังกฤษมากนัก กล่าวคือ

1. ความกว้างของแพตเทิร์นขนาด 8 บิต น้อยไปสำหรับอักษรไทยบางตัว เช่น ฒ

2. ความสูงของแพตเทิร์นขนาด 8 บิต น้อยไปสำหรับอักษรไทยบางตัว เช่น ไ และ ฐ

3. อักษรไทยต้องแสดงอย่างน้อยถึง 3 ระดับ คือ กลาง ล่าง และ บน

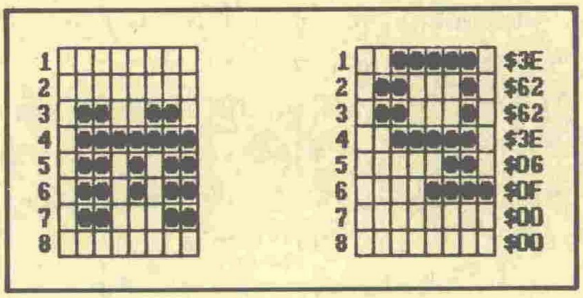
การแก้ปัญหา

ในการแก้ปัญหานี้ผมเริ่มด้วยการพิจารณาว่าความกว้างของแพตเทิร์นขนาด 8 บิต ซึ่งถึงแม้จะน้อยไปสำหรับภาษาไทยบางตัวแต่ก็ยอมรับ (จะทำให้ได้ตัว ฒ ที่อ่านยากและไม่สวย หากต้องการแก้ปัญหานี้ก็ทำได้ครับ และผมจะนำไปกล่าวในบทความตอนต่อไป)

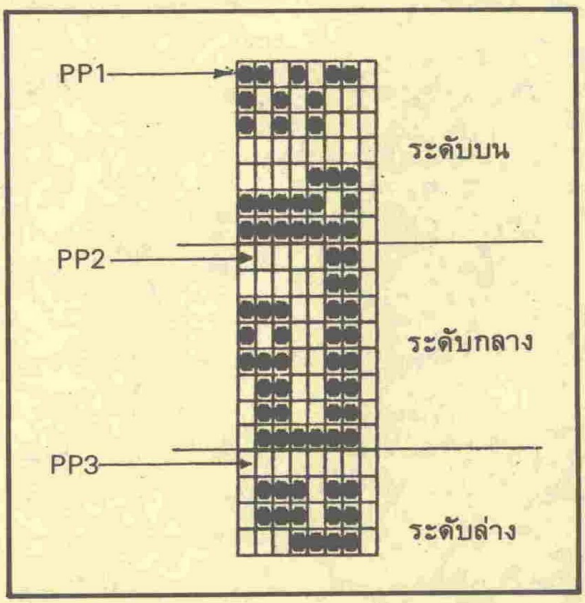
ส่วนความสูงของแพตเทิร์นขนาด 8 บิต น้อยไปสำหรับภาษาไทยบางตัว ผมขอพักไว้ นำเรื่องที่อักษรไทยต้องแสดงหลายระดับมาพิจารณาก่อน ซึ่งผมขอรวบรัดตัดตอนเป็นดังแสดงในรูปที่ 4 เป็นการกำหนดเส้นสแกนไลน์ให้แก่ตัวอักษรแต่ละระดับ โดยกำหนดให้ระดับบนใช้ 7 เส้น ระดับกลางใช้ 8 เส้น ระดับล่างใช้ 4 เส้น รวม 19 เส้น หรือคือตัวหนังสือภาษาไทยทุกระดับจะแสดงอยู่ในกรอบอักขระขนาด 8×19 จุดภาพ ใน 1 จอภาพจะแสดงได้ 18 บรรทัด

จะเห็นว่าได้จำนวนบรรทัดต่อ 1 จอภาพน้อยไป ที่จะลดได้คือระดับบน เช่นลดเป็น 5 จะได้ 20 บรรทัด แต่จะทำให้ภาพอักขระที่ระดับบนไม่ชัด

สำหรับปัญหาเรื่องความสูงของอักษรไทยบางตัวเช่น ไ ผมคิดจะให้ขึ้นไปใช้เนื้อที่ของสระบน ซึ่งไม่มีปัญหาใด ส่วนอีกบางตัว เช่น ฐ จะให้ลงไปใช้เนื้อที่ของสระล่างซึ่งจะมีปัญหาเกิดขึ้นเพราะอักษรพวกนี้บางตัวมีสระล่าง เช่น คำว่า "กระฎุมพี" ก็จะต้องยอมรับครับว่าจะต้องมีการทับกันของตัวอักษรเกิดขึ้นบ้าง



รูปที่ 3 แสดงแพตเทิร์นของอักษรที่เล็ก m และ q



รูปที่ 4 แสดงการจัดแบ่งระดับของอักขระ

จัดแบ่งโปรแกรม

เมื่อได้กำหนดรูปแบบการแสดงภาพอักษรแล้ว ปัญหาต่อไปคือ จะจัดแบ่งหน้าที่ให้แก่โปรแกรมต่าง ๆ อย่างไร เช่น โปรแกรมตามรูปที่ 1 จะนำภาพอักขระไปออกจอภาพตรงจุด PP (อยู่ตรงมุมบนซ้ายของกรอบภาพ) โดยไม่คำนึงว่า จะเป็นอักขระบน กลาง หรือล่าง

ในเท็กซ์โหมดโปรแกรมที่ทำหน้าที่นี้มีอยู่แล้ว และเราไม่มีทางเลือก จึงต้องเขียนโปรแกรมขึ้นใหม่เพื่อเลื่อนเคอร์เซอร์ไปมา เพื่อให้แสดงอักษรไทยได้ 3 ระดับ แต่ในกราฟิกโหมดผมมี

แนวความคิดว่า จะให้โปรแกรมที่จะดำเนินการในเรื่องนี้และขอตั้งชื่อว่า OutTch เป็นผู้ดำเนินการจัดระดับด้วย

การจัดระดับ

ขอให้ย้อนกลับไปดูรูปที่ 4 ครับ ผมได้เขียนตำบลที่ของจุด PP ไว้ 3 แห่ง คือ PP1, PP2 และ PP3 ในการจัดระดับเราจะต้องคำนึงก่อนว่าจะให้จุด PP อยู่ที่ตำแหน่งใดจึงจะเหมาะสม

ความจริงผมคิดอยู่หลายรอบถึงข้อดีข้อเสีย ผลที่สุดเห็นว่าอยู่ตรงจุด PP1 จะเหมาะสมกว่าโดยที่ ถ้าต้องการแสดงภาพ อักขระบนให้แสดงได้โดยตรง ถ้าเป็นอักขระกลางให้เลื่อนจุด PP มาที่ PP2 แล้วแสดง (เว้นอักขระบางตัวดังจะกล่าวต่อไป) ถ้าเป็นอักขระล่างให้เลื่อนจุด PP มาที่ PP3 แล้วแสดง

ซึ่งในเท็กซ์โหมดเราไม่มีทางเลือก จึงต้องดำเนินการโดยตรวจรหัสอักขระโดยตรง เช่น

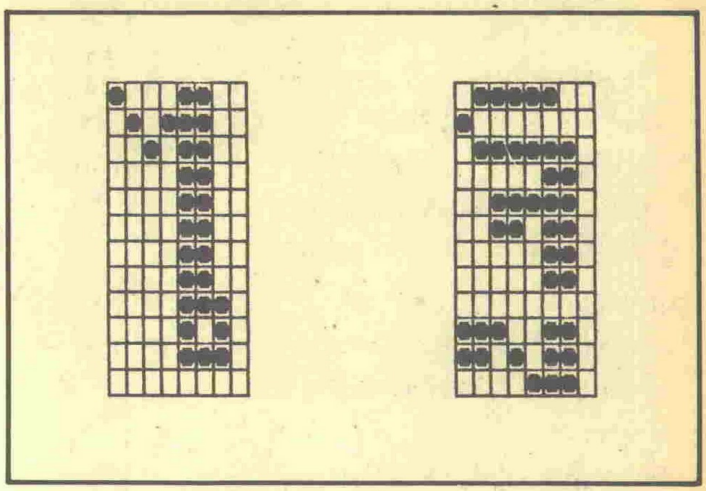
IF ch < = 'ฮ' THEN MoveTo (PP2, y)

แต่ตรงไหนมี IF ตรงนั้นคือเวลาที่จะต้องเสียไป ผมจึงมีแนวความคิดว่าจะบวกค่า PP ในแกน y คือ prey ด้วยค่า 0,7 และ 15 สำหรับอักขระในระดับบน กลาง และล่างตามลำดับ โดยไม่ตรวจสอบ (เพราะตรวจสอบแล้วบวกหรือไม่บวก จะเสียเวลามากกว่าบวกโดยไม่ตรวจสอบ)

ซึ่งจะทำได้โดยเก็บค่านี้ไว้ในแพตเทิร์นของอักขระแต่ละตัวด้วย และขอเรียกค่านี้ว่า dify แต่การดำเนินการจริงจะไม่บวกค่า dify เข้ากับและให้ prey จะใช้ค่า prey + dify แทน

จุด PP กับการตกบรรทัด

ในเท็กซ์โหมดโดยเฉพาะกับภาษาอังกฤษเมื่อเราป้อนอักขระใดไปแล้ว เคอร์เซอร์จะเลื่อนไปอยู่หลังอักขระนั้น



รูปที่ 5 แสดงการสร้างอักษรที่มีความสูงเกินกว่าปกติ สระไอตามรูปสูงเพียง 11 จุด เพื่อให้เห็นแตกต่างกัน ค่าที่เหมาะสมคือ 12 เช่นกัน

และถ้าสุดบรรทัดคือตกบรรทัด จะขึ้นบรรทัดใหม่โดยอัตโนมัติโดยไม่มีโปรแกรมใดดำเนินการ เพราะหน่วยความจำเรียงติดต่อกัน ส่วนภาษาไทยจะต้องเขียนโปรแกรมให้เคอร์เซอร์ขึ้นบรรทัดใหม่ และกลับไปเมื่อผู้ใช้กดคีย์อักขระบนหรือล่าง

สำหรับในกราฟิกโหมด ผมมีแนวความคิดว่าจะไม่ให้ OutTch ดำเนินการในเรื่องนี้ เพื่อไม่ต้องมี IF ดังกล่าวมาและสามารถดำเนินการได้ใน ReadTStr กับ WriteTStr ดังนั้นเมื่อให้ OutTch อักขระในระดับกลางแล้วจะคงให้จุด PP อยู่ ณ กรอบหน้าต่างเดิมจนกว่าจะ OutTch ด้วยอักขระกลางตัวใหม่

ซึ่งจะทำได้โดยบวกค่า PP ในแกน x คือ prex ด้วยค่า 0, 8 และ 0 สำหรับอักขระ บนกลาง และล่างตามลำดับ เรียกค่านี้ว่า difx และเก็บไว้ในแพตเทิร์นของอักขระแต่ละตัวเช่นกัน

ในการนี้จะต้องให้ค่า prex เมื่อเริ่มต้นมีค่าเป็น -8 ของตำบลที่ PP จริง เช่น หากต้องการให้จุด PP อยู่ที่ (0,0) จะต้องให้ค่า prex เป็น -8 หรือถ้าต้องการให้อยู่ที่จุด (10,20) จะต้องให้ค่า prex เป็น 2 แต่ค่า 8 ที่กล่าวมานี้ที่ใช้จริงเป็น 9 เพื่อเว้นช่องว่างระหว่างอักขระอีก 1 จุดภาพ

ปัญหาเรื่องความสูง

ถึงตรงนี้ผมก็สามารถแก้ปัญหาเรื่องความสูงที่ค้างไว้ได้แล้วครับ คือ ถึงแม้เราจะให้อักขระกลางบางตัวสูงกว่า 8 ได้แต่ก็ต้องไม่เกิน 12 เพราะอักขระกลางบางตัวเช่น ฏ ใช้เนื้อที่ของอักขระล่างซึ่งมีอยู่เพียง 4 เส้นสแกนไลน์ ผมจึงกำหนดไว้ไม่เกิน 12 เส้นสแกนไลน์

เพื่อรวบรัดขอให้ดูในรูปที่ 5 แสดงถึงการสร้างอักขระ ไ และ ฐ ในความสูงไม่เกิน 12 เส้น นั่นคืออักขระแต่ละตัวจะใช้หน่วยความจำไม่เกิน 12 ไบต์สำหรับเก็บแพตเทิร์นของตน แต่ผมจะให้เป็น 16 ไบต์เท่ากันหมดเพื่อเก็บค่า difx, dify และค่าอื่นอีกดังจะกล่าวต่อไป

จากที่ความสูงของอักขระแต่ละตัวไม่เท่ากันเช่นนี้ ผมจึงให้เก็บค่าความสูงไว้กับแพตเทิร์นอีก เรียกว่าค่า high เพื่อให้โปรแกรม OutTch ซึ่งมีการทำงานคล้ายโปรแกรมในรูปที่ 1 ใช้แทนค่า 7 คือแทนที่จะแสดงอักขระทุกตัวในแบบ 8 เส้นสแกนไลน์เท่ากันหมด จะแสดงเท่ากับความสูงของอักขระตัวนั้น ๆ ทำให้ประหยัดเวลาสำหรับอักขระที่เตี้ยกว่า 8 เส้นและสามารถแสดงอักขระที่สูงกว่า 8 เส้นได้

แต่ทั้งนี้ต้องประกอบด้วยการพิจารณาต่อไปนี้ด้วย คือ อักษรที่ไม่มี

อักขระ	difx	dify	high
ตัวเลขและเครื่องหมายต่าง ๆ	9	7	7
อักษรภาษาอังกฤษตัวใหญ่	9	8	6
อักษรภาษาอังกฤษตัวเล็ก	9	10	4
อักขระกลาง ธรรมดา เช่น ก	9	9	5
อักขระกลาง มีหาง เช่น ป	9	7	7
อักขระกลาง สูง เช่น ไ	9	3	11
อักขระกลาง มีเชิง เช่น ฎ	9	9	9
อักขระกลาง มีเชิงมีหัว เช่น ฐ	9	7	11
อักขระบน เดี่ยว เช่น ่	0	2	3
อักขระบน ผสม เช่น ะ	0	0	6
อักขระล่าง เช่น สระอุ	0	16	2

รูปที่ 6 ค่าที่จะให้แก่แพตเทิร์นของอักขระแต่ละตัว ค่า high นับจาก 0 จึงเป็นค่าที่น้อยกว่าความสูงจริงอยู่ 1 พวกตัวเลขและเครื่องหมายต่าง ๆ กำหนดไว้ในแบบ 8×8 ที่บรรทัดกลาง ควรปรับให้เหมาะสมแต่ละตัว และความจริงค่าที่ให้นี้เป็นค่าตัวอย่างอาจต้องมีการปรับปรุงให้เหมาะแก่อักขระแต่ละตัวเมื่อสร้างแพตเทิร์น

หางหรือฐาน เช่น ก ข ค ย ควรจะให้แสดงกี่เส้น คำตอบคือ 6 เส้นเพราะถ้าเตี้ยกว่านี้จะแสดงอักษรพวก ค ด ได้ไม่ชัดเจน ถ้าให้สูงกว่านี้ก็จะไม่มีที่สำหรับหางของตัว ป ศ ส

เพื่อประหยัดเวลาดังกล่าวมา การสร้างแพตเทิร์นของอักษรต่าง ๆ เช่น อักษร ก เราจะไม่เว้นที่ข้างบน ดังนั้นในการแสดงภาพจึงต้องเลื่อนจุด PP ลงอีก 2 เส้น นั่นคือต้องปรับค่า dify ดังกล่าวมาของอักษร แต่ละกลุ่มเป็นดังตารางในรูปที่ 6 สำหรับค่า high ต้องน้อยกว่าที่เป็นจริง 1 เพราะจะเริ่มวงรอบ FOR ด้วยค่า 0

แบบของข้อมูล

จากเรื่องราวที่กล่าวมานี้สามารถนำมากำหนดแบบข้อมูลสำหรับแพตเทิร์นได้ดังรูปที่ 7 ที่ผมให้ OneCharT เป็นอะเรย์ขนาด 16 ช่อง ทั้งที่ใช้จริงเพียง 15 ช่องก็เพื่อให้ลงตัว และหากต่อไปจะกำหนดความกว้างของอักขระเพิ่ม ส่วน ThCharT จะเป็นอะเรย์ขนาด 256 ช่องของ OneChart เพื่อเก็บแพตเทิร์นของทั้งอักษรไทยและอักษรอังกฤษ

```
CONST difX = 12;  difY = 13;  high = 14;
TYPE  OneCharT = ARRAY[0..15]  OF  Byte;
      ThCharT  = ARRAY[Char] OF OneCharT;
VAR   preX,preY : Integer;  cc : Word;
      tfont : ThCharT;


PROCEDURE  OutTch(ch : Char);
TYPE   WeighT = ARRAY[0..7] OF Byte;
CONST  map : WeighT =
       ($80,$40,$20,$10,$08,$04,$02,$01);
VAR    x,y,py : Integer;
BEGIN
  preX := preX + tfont[ch,difX];
  py := preY + tfont[ch,difY];
  for  y := 0 to tfont[ch,high]  do
    for  x := 0 to 7  do
      if  tfont[ch,y] and map[x] <> 0  then
        PutPixel(preX+x, py+y, cc);
END;
```

รูปที่ 7 แบบข้อมูลและโพรซีเดอร์ OutTch

โพรซีเดอร์ OutTch

ได้เขียนโพรซีเดอร์ OutTch เพื่อให้เห็นเป็นแนวทางว่า จะนำแพตเทิร์นของอักขระตามที่กล่าวมานี้ไปออกจอภาพได้อย่างไร ไว้ในรูปที่ 7 เช่นกัน

จากโปรแกรมนี้จะเห็นว่าจะต้องมีโปรแกรมดำเนินการให้ค่า prex, prey และ cc ไว้ก่อน ซึ่ง cc เป็นค่าสีเพื่อให้สามารถแสดงอักขระแต่ละตัวให้เป็นสีต่าง ๆ ได้

เท่าที่กล่าวมานี้เป็นการทำภาษาไทยในกราฟิกโหมดทางด้านเอาต์พุต แต่ยังไม่สมบูรณ์เนื่องจากยังไม่มีแพตเทิร์นของอักขระ ในฉบับหน้าจะนำเรื่องการสร้างแพตเทิร์นมาเสนอครับ